REMARKS

Claims 1-16 are pending in this application. In this Response, Applicants have provided remarks that explain some of the differences between the present invention and the reference cited by the Examiner. In light of these differences, Applicants submit that the present application is in condition for allowance.

THE REJECTIONS UNDER 35 U.S.C. § 102

At pages 2-4 of the Office Action, the Examiner rejected claims 1-16 under 35 U.S.C. § 102(e) based on U.S. Patent No. 6,345,272 to Witkowski *et al.* ("Witkowski"). Applicants provided a detailed description of the system disclosed by Witkowski in the Response filed on September 29, 2006. The arguments set forth in that Response are maintained herein. In an attempt to facilitate allowance, Applicants submit the following arguments to further clarify the differences between Witkowski and the present invention.

Witkowski explicitly states that the purpose of the invention disclosed therein is to provide a method and apparatus for rewriting aggregate queries to access a materialized view. *See* abstract. To accomplish this, Witkowski generates three separate queries: (i) an aggregate query 210; (ii) a materialized view query 270; and (iii) a query 280 that accesses the materialized view. Each of these three types of queries is discussed below.

The aggregate query 210 requests summary information from a first table, *e.g.,* sales table 250. *See* Col. 4, lines 7-16. This query aggregates the data in the sales table 250, which contains many different columns of data. *Id.* Witkowski also discloses a materialized view query 270, that aggregates data from the "$AMT" column of the sales table 250. The result of the materialized view query 270 is a second table MVSales, that includes a column titled "Sum_Sales." Thus, Witkowski discloses two tables, *e.g.,* a sales table 250 and a MVSales table.

Witkowski teaches that it is often desirable to access a materialized view table, *e.g.,* MVSales, rather than a table with several columns, *e.g.,* sales table 250, for various reasons. *See* Col. 1, lines 11-20. In order to do this, Witkowski teaches that aggregate query 210 may be rewritten to access the materialized view table MVSales instead of the sales table 250. *See* Col. 3, line 66 - Col. 4, line 47. This is performed by rewriting aggregate query 210 as query 280. *Id.*

By rewriting the query, aggregate query 280 performs an aggregation of the "Sum_Sales" column of the MVSales table only. In this manner, the first sales table 250 is not referenced by

query 280. Rather, query 280 accesses only the second table MVSales. Indeed, query 280, illustrated in FIG. 2, only references the "Sum_Sales" column of the MVSales table created by materialized view query 270. *See* FIG. 2. Applicants argument is further supported by the fact that there is no reference to the sales table 250 in query 280.

It is important to note that the three queries disclosed by Witkowski are separate and distinct. In other words, query 210 is one way of creating a table by accessing the first sales table 250. Query 270 is a separate query that creates a materialized view by accessing the first sales table 250. In contrast to these two queries, query 280 does not access first sales table 250. Instead, it is a separate query that aggregates data only from a second table created by materialized view query 270.

In the Office Action, the Examiner repeatedly contends that query 280 joins the data from table 250 and the summary table MVSales. The Examiner provides no support for this contention, however, and the argument lacks basis. It appears that the Examiner's contention stems from the fact that the "Sum_Sales" column is a result of the aggregation of data from table 250. However, the mere fact that the "Sum_Sales" column is a result of aggregation of data from table 250 does not mean that query 280 implements a join of the two tables. Rather, query 270 aggregates the data from table 250, at which point it is stored as a separate table. *See* Col. 1, line 64 - Col. 2, line 6. When desired, query 280 aggregates the data from the "Sum_Sales" column (recall this is stored separately) without any reference to, or consideration of, table 250. *See* FIG. 2. Thus, query 280 only accesses one table, and thus cannot logically implement a join. *Id.*

Another way of explaining the error in the Examiner's interpretation is as follows. Both Applicants and the Examiner can agree that sales table 250 is a separate table. The manner in which sales table 250 is formed, *e.g.*, the specific query, is of no consequence. Moreover, query 210 aggregates data from sales table 250, and a skilled artisan would not assert that query 210 performs a join operation. The function of query 280 is tantamount to the function of query 210. That is, the MVSales table is stored separately in a memory. *See* Col. 1, line 64 - Col. 2, line 6. Since it is a separate table, the manner in which MVSales is formed is of no consequence. Further, query 280 simply aggregates a row, *e.g.*, "Sum_Sales," from the MVSales table. Thus, a skilled artisan would not assert that query 280 performs a join operation.

In contrast, the present invention, for example as recited in claim 1, comprises a query generator for generating a query for obtaining selected data from a database. The database has a

number of detail tables in which data is stored, and the query generator comprises a processor that is coupled to the database in use. The processor is adapted to receive an input indicating the selected data to be obtained to generate a first query. The input is then analyzed to determine whether the input requires a joining of data in a plurality of different detail tables, and an aggregation step. If it is determined that the input requires a joining of data in a plurality of different detail tables, the processor modifies the content of the first query indicating the selected data to be obtained to generate a second query. The second query is adapted to cause the database to generate a query that aggregates data within each of a plurality of detail tables. Then, a SQL join operation is performed to join the aggregated data from each of the plurality of detail tables.

As mentioned above, Witkowski does not perform a SQL join operation because it accesses only one table, *i.e.*, the summary table called "Sum_Sales." Thus, Witkowski does not teach each and every element of the present invention as recited in independent claims 1, 9, and 16. As such, Applicants submit that the Examiner's § 102(e) rejections have been overcome. Reconsideration and allowance of the pending claims is respectfully requested.

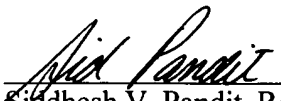
CONCLUSION

All claims are believed to be in condition for allowance. If the Examiner believes that the present remarks still do not resolve all of the issues regarding patentability of the pending claims, Applicants invite the Examiner to contact the undersigned attorneys to discuss any remaining issues.

A Petition for Extension of Time is submitted herewith extending the time for response two months to and including March 19, 2007. No other fees are believed to be due at this time. Should any fee be required, however, please charge such fee to Bingham McCutchen LLP Deposit Account No. 195127, Order No. 19111.0045.

Respectfully submitted,
BINGHAM MCCUTCHEN LLP

Dated: March 15, 2007

By: 

Siddhesh V. Pandit, Registration No. 58,572
BINGHAM MCCUTCHEN LLP
2020 K Street, NW
Washington, D.C. 20006
(202) 373-6513 Telephone
(202) 373-6440 Facsimile